
UNIVERSITÀ DEGLI STUDI DI PISA

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

Tesi di Laurea

# Blockchain: tra Altcoin e nuove applicazioni

Relatore:
Prof. **Anna Bernasconi**

Candidato:
**Alessandro Pagiaro**

Correlatore:
Dott. **Damiano Di Francesco Maesa**

ANNO ACCADEMICO 2015-2016

# Indice

# Introduzione

Nel 2009 una persona o un gruppo di persone dietro lo pseudonimo di Satoshi Nakamoto ha inventato la moneta digitale del nuovo secolo: i Bitcoin. Anche se molto spesso si sente parlare sulla stampa generica di questa novità, non così tanto spesso si sente parlare della tecnologia che ne permette il funzionamento. La principale novità tecnologica su cui si fonda questa moneta è la Blockchain. Sarà proprio interesse di queste pagine approfondirne la struttura e ragionare su utilizzi alternativi a quelli per cui è stata creata, cioè il salvare le transazioni di Bitcoin. Molti, di fatto, sostengono che Bitcoin sia destinato a fallire come progetto in quanto non capace di rispondere alla rapidità richiesta per gli scambi al giorno d'oggi e per l'eccessiva incertezza che l'utilizzo ne comporta. Quello invece su cui molto si discute negli ultimi tempi è sull'evoluzione che la Blockchain potrà portare ai servizi del web e come questi saranno più trasparenti e sicuri. Non sarà certo possibile trattare in queste poche pagine tutti i mille utilizzi che sono nati o che sono stati proposti, da reti peer-to-peer come ZeroNet, a social network come Alaska, a servizi DNS come i Namecoin o ad applicazioni per gli *smart contract* come Ethereum. Si cercherà tuttavia di studiarne la struttura per capirne le potenzialità lasciando poi libertà al lettore di approfondire e, magari, inventare nuovi utilizzi consapevoli degli ingranaggi che ne permettono il funzionamento.

La relazione è organizzata in tre parti.

La prima cerca di offrire una panoramica veloce su Bitcoin, spiegandone il funzionamento. Attraverso l'esempio di un pagamento andremo ad approfondire tutte le strutture dati che sono state impiegati per la realizzazione dell'intero sistema. Si andrà a vedere come queste monete riescono a gestire il consenso decentralizzato tra tutti gli utenti e come sia affidabile il sistema di controllo e gestione delle transazione.

Nella seconda parte verrà spiegato come la Blockchain opera, fornendo esempi di verifica e validazione dei blocchi (*mining*) differenti dal classico Proof-of-Work impiegato in Bitcoin. Tali metodi risulteranno spesso più veloci ma non ne è ancora dimostrata l'efficacia su una rete grande come quella dei Bitcoin e sono spesso ancora in fase di studio.

Infine, nell'ultima parte della relazione si discuteranno delle proposte per utilizzo della Blockchain differenti dalla moneta digitale. Essendo un

campo di studi piuttosto nuovo, la presentazione risulterà priva di dettagli implementativi specifici, ma cercherà, invece, di fornire una panoramica su come vari servizi possono essere resi gratuiti e liberi con questa tecnologia.

# Capitolo 1

# Bitcoin

Bitcoin è un sistema di pagamento virtuale che sfrutta la crittografia per permettere agli utenti di effettuare transazioni in maniera totalmente decentralizzata. Il protocollo su cui si basa può essere eseguito su qualsiasi macchina, è *open source* ed è installato ad oggi su una grande quantità di dispositivi. Fu introdotto dalla geniale idea di Satoshi Nakamoto [Macci, 2016]. Il termine Bitcoin [1] (BTC) rappresenta l'unità di valuta virtuale scambiata tra i partecipanti al sistema. Gli utenti possono trasferire denaro effettuando delle transazioni che vengono gestite da alcuni programmi (detti *wallet*) in maniera del tutto simile ai normali bonifici bancari. È possibile vendere e comprare beni reali tramite questa moneta, scambiarla con altre valuta o finanziare progetti. Tuttavia, a differenza delle monete tradizionali, Bitcoin non prevede alcun tipo di banconota e l'intero sistema viene gestito solo attraverso transazioni digitali. Effettivamente i coins vanno immaginati come un insieme di transazioni che un utente ha ricevuto ed è quest'insieme che compone il saldo finale.

Il sistema Bitcoin è un sistema distribuito, peer-to-peer, aperto a tutti dove non esiste un *server* o un controllo centralizzato di alcun tipo. Nuovi bitcoin vengono generati dagli utenti tramite un processo chiamato *mining* che consiste nel risolvere un problema matematico sufficientemente complesso, proteggendo il protocollo da alcuni attacchi. Il sistema prevede una gestione della complessità del problema in relazione alla capacità di calcolo della rete, riuscendo a mantenere probabilisticamente costante il tempo di *mining*. La quantità di valuta coniata dal processo di *mining* viene inoltre dimezzata regolarmente, cosa che porterà ad un limite massimo di Bitcoin in circolazione nel 2140, quando saranno stati minati 21 milioni di Bitcoin (BTC), riuscendone così a controllare l'inflazione.

[1] Con l'iniziale maiuscola ci riferiremo alla moneta introdotta da Satoshi Nakamoto e al protocollo che la supporta, con il termine bitcoin con l'iniziale minuscola all'insieme di valute virtuali esistenti (Bitcoin compresi) e la comunità che ne fa uso.

Da un punto di vista più tecnico, Bitcoin è un insieme di protocolli e algoritmi che coesistono per la realizzazione dell'intero sistema. Struttura centrale ed innovativa di Bitcoin è la **Blockchain**, un database distribuito e decentralizzato che regola lo scambio di valuta dell'intero sistema. Sarà compito di questo database, la Blockchain, tenere traccia e verificare la validità delle transazioni, passate e future, e permettere di dare una risposta pratica al problema del consenso distribuito (il Problema dei Generali Bizantini).

## 1.1 Storia di una transazione

Per spiegare con facilità il funzionamento dell'intero sistema bitcoin ci faremo aiutare da un esempio. Vediamo come Alice può effettuare un pagamento a Bob e come Cloe possa validare la transazione.

Alice vuole acquistare un oggetto da Bob. Alice e Bob, come ogni altro utente del sistema, disporranno di una *chiave pubblica* e di una *chiave privata* che chiameremo da ora in poi, rispettivamente $K_{pu_A}$ e $K_{pr_A}$ dove A ci indica che ci riferiamo ai valori di Alice (B nel caso di Bob, C nel caso di Cloe).

Per prima cosa Alice deve conoscere l'identificativo di Bob, un indirizzo che viene generato a partire da $K_{pu_B}$. Tale valore può essere liberamente pubblicato da Bob sul canale che preferisce in quanto la sua pubblicazione non indebolisce la sicurezza del sistema. Alice, presa nota dell'identificativo di Bob, costruisce quindi un messaggio che rappresenta una transazione secondo lo schema mostrato nella Tabella 1.1

### 1.1.1 Schema di una transazione

| Dimensione | Campo | Descrizione |
|---|---|---|
| 4 bytes | Versione | Specifica quale versione del protocollo usa la transazione |
| 1-9 bytes | Contatore di Input | Indica quanti input sono dati |
| Variabile | Input | Una o più input |
| 1-9 bytes | Contatore di Output | Indica quanti output sono dati |
| Variabile | Output | Uno o più output |
| 4 bytes | Locktime | Lo Unix timestamping o il numero di blocco |

Tabella 1.1: Struttura di una transazione

Come è chiaro guardando la struttura dati di una transazione, per effettuare un pagamento occorre fornire un insieme di input e un insieme di output. La differenza tra la somma dei valori in input e quelli in output deve essere

$\geq 0$ [2]. Gli input sono scelti da Alice dall'insieme di transazioni ricevute in passato. Tale set è denominato *UTXO* (Unspent Transaction Output).

Supponiamo che Alice voglia inviare 5.5 BTC a Bob e che l'UTXO di Alice sia il seguente:

| ID Transazione | Valore |
|---|---|
| 101 | 1 BTC |
| 203 | 3 BTC |
| 465 | 5 BTC |
| ... | ... |

Alice dovrà quindi scegliere dal suo insieme UTXO alcune transazioni da inserire nell'input della transazione che sta andando a creare, ponendo l'attenzione sulla somma dei valori delle transazioni scelte che deve risultare superiore a 5.5 BTC, cioè la somma che deve pagare a Bob. Scelte quindi le transazioni, nel nostro esempio quelle con ID pari a 465 e 101, la somma dei valori in input viene calcolata risultando di 6 BTC. Alice inserisce quindi in output due nuove transazioni, una dal valore di 5.5 BTC indirizzata a Bob e una del valore di 0.5 BTC indirizzata a se stessa, che possiamo vedere come il resto.

### 1.1.2 Propagazione di una transazione

Finita questa fase di preparazione, la transazione viene firmata da Alice tramite il protocollo Elliptic Curve Digital Signature Algorithm (ECDSA) e quindi spedita a tutti nel network (una rete di nodi p2p completamente decentralizzata) ed inserita nell'insieme delle transazioni non ancora verificate. Mandare questo messaggio non è un operazione molto complessa in quanto il messaggio non ha bisogno di viaggiare cifrato poiché non contiene informazioni confidenziali, non è necessario verificare la fonte poiché la transazione è firmata e non è necessaria una infrastruttura specifica poiché può essere scelto il mezzo di trasmissione che si preferisce. Ogni nodo, quando riceve la transazione, se la riconosce valida la inoltra a tutti i suoi vicini che ancora non l'hanno ricevuta. Tale processo di propagazione viene chiamato *flooding*.

### 1.1.3 Inserimento della transazione nella Blockchain

A questo punto Bob riceve da Alice la comunicazione che il pagamento è stato effettuato. Tuttavia Bob non ha alcun modo di verificare che le informazioni che Alice ha dichiarato nel messaggio, cioè le transazioni in input, siano valide e legittime e neanche che Alice non abbia contemporaneamente

[2] A volte tale differenza può non essere nulla, in tal caso la differenza verrà prelevata dal *miners* e sarà considerata commissione della transazione

pubblicato due transazioni con gli stessi valori in input, andando di fatto a spendere due volte la stessa somma. Per questo interviene una terza persona, Cloe, che si preoccuperà di verificare un insieme di transazioni e certificarne la validità. Il processo di verifica che svolge Cloe è detto *mining* e chi lo effettua è chiamato *miner* (si veda la sezione 2.2 per approfondire).

Appena Cloe riuscirà a certificare il blocco e la fase di mining sarà conclusa, il blocco verrà inviato a tutti i nodi del network e sarà definitivamente aggiunto alla Blockchain. A questo punto Bob può considerare la transazione di Alice conclusa con successo (si approfondirà maggiormente questa fase nel Capitolo 2).

# Capitolo 2

# Blockchain

La Blockchain è lo strumento innovativo che ha permesso di dare una soluzione pratica ad un antico problema dell'informatica, il Problema dei Generali Bizantini. Il problema consiste nel riuscire a mettere d'accordo un numero $n$ di entità, nel caso di Bitcoin si tratta di utenti, sul valore di un oggetto. La difficoltà è dovuta al fatto che possono esserci all'interno del sistema entità scorrette che propagano un valore errato. La soluzione deve prevedere questa opzione e risolvere riuscendo a far emergere il valore corretto. Analizzeremo in questa prima parte del capitolo il modo in cui la Blockchain è impiegata in Bitcoin e discuteremo poi utilizzi alternativi.

## 2.1 La Blockchain in Bitcoin

### 2.1.1 La struttura

La Blockchain è una catena di blocchi con puntatore al blocco precedente. Tale valore altro non è che il risultato della funzione hash [1] del blocco $i-1$, che viene usato come identificativo univoco, salvato in un campo predisposto nell'header nel blocco $i$. L'hash viene usato anche come identificatore del blocco. Questo meccanismo di precedente inclusione ci garantisce la possibilità di aggiungere nuovi blocchi senza dover modificare i precedenti ma allo stesso tempo essere certi che la modifica di un blocco precedente ci comporta la modifica a catena di tutti i blocchi successivi in quanto il valore del puntatore al blocco precedente va modificato.

### 2.1.2 Il blocco

La struttura tipica di un blocco nella catena è rappresentata nella Figura 2.1 e contiene i seguenti campi:

[1] La funzione hash applicata è la *double-SHA256*

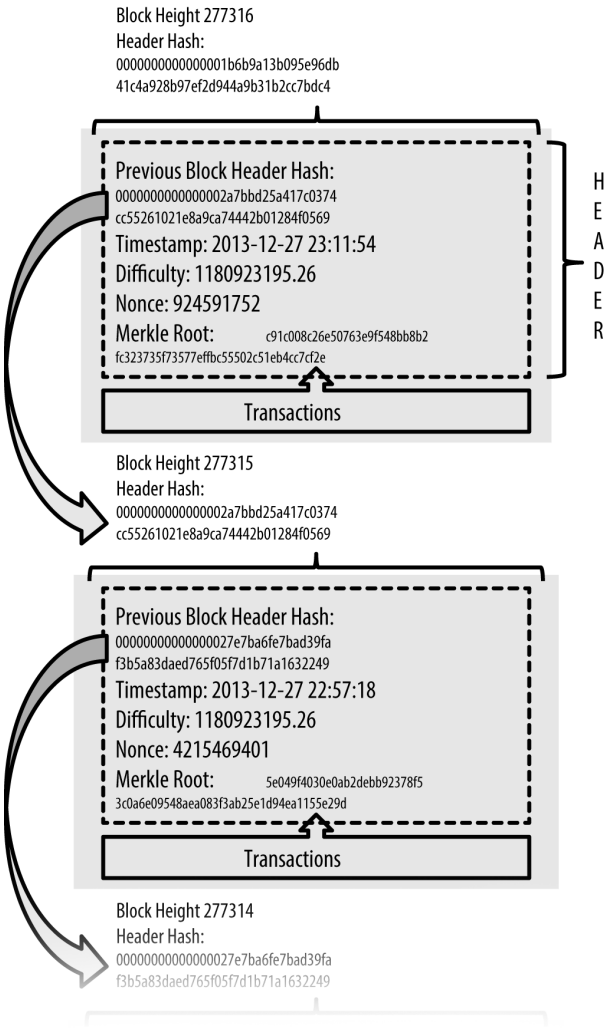
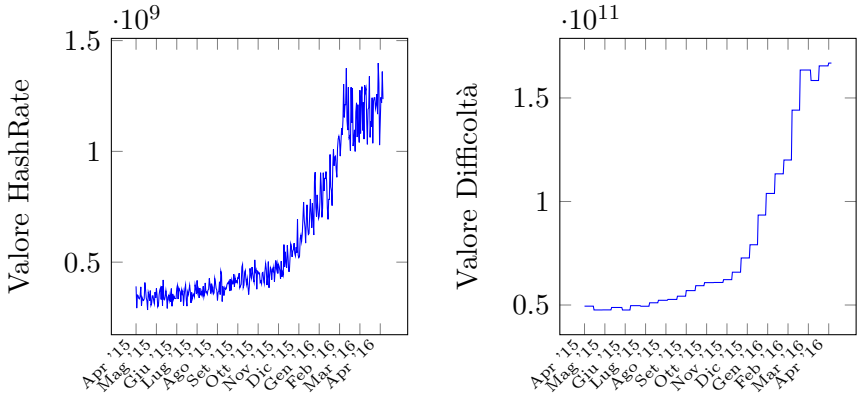



Figura 2.1: Stuttura della Blockchain, da [1]

### Difficoltà

Il campo *Difficulty* rappresenta la difficoltà richiesta per il problema del Proof-Of-Work (Capitolo 3) per il blocco.

Il primo blocco della catena presenta un indice di difficoltà pari ad 1.0. Questo valore viene ricalcolato ogni 2016 blocchi (orientativamente due settimane se si considera che un blocco richiede circa 10 minuti per la fase di




(a) Grafico valore dell'hashrate complessivo della rete

(b) Grafico valore della Difficoltà

*mining*) secondo la seguente funzione

$$New\ Difficulty = Old\ Difficulty \cdot \frac{Actual\ Time\ of\ Last\ 2016\ Blocks}{20160\ minutes}$$

Tale formula ci permette di mantenere costante il tempo di *mining* di un blocco, intorno ai 10 minuti, anche all'incrementare della potenza di calcolo del nuovo hardware prodotto [2].

I due grafici riportati mostrano l'hash rate (cioè la quantità di hash che l'intera rete di nodi è in grado di calcolare in un secondo) e il fattore di difficoltà richiesto dal sistema in relazione all'hash rete nel corso di un anno [3] evidenziando come al variare dell'hash rate viene modificato il valore del fattore di difficoltà.

### Nonce

Il *nonce* è il seme con cui viene gestita la Proof-of-Work di cui si parlerà in maniera più estesa nel Capitolo 3. Il *nonce* è un intero che viene concatenato al blocco al momento del calcolo dell'hash per garantire una regolarità nell'hash stesso. Viene calcolato per tentativi successivi e all'aumentare della difficoltà ci si aspetterà, vista la distribuzione della funzione hash *SHA256* impiegata, di dover effettuare un maggior numero di tentativi per la risoluzione del problema e di dover provare, quindi, un maggior numero di *nonce*.

[2] Il valore 2016, che compare nella formula del calcolo della difficoltà, è stato scelto alla creazione del sistema ed è quindi una costante.

[3] Dal 1 Aprile 2015 al 1 Aprile 2016

**Merkle Root**

Questo campo contiene il valore della radice dell'albero di Merkle che contiene un hash derivante dalla concatenazione degli hash di tutte le transazioni incluse nel blocco (si veda l'Appendice, sezione 5.1).

## 2.2 Mining di un blocco

Vediamo ora il processo di *mining* che porta alla convalida di un blocco permettendone la sua aggiunta alla Blockchain. Il processo è attuabile da qualsiasi nodo che partecipa alla rete purchè abbia una quantità sufficiente di risorse per gestirne la computazione. Occorre, difatti, una discreta quantità di memoria per mantenere in locale le transazioni non ancora spese e, a seconda del metodo di convalida usato, può essere necessario disporre di una elevata quantità di risorse computazionali o di memoria principale.

Non ci dilungheremo troppo sul metodo di Proof-of-Work (che sarà tema centrale del Capitolo 3) ma ci concentriamo sulle operazioni che deve effettuare il *miner*, cioè il modo in cui si avvia un processo di *mining*, per verificare la validità di un blocco.

Il processo di *mining* è un processo fondamentale e per incentivare i nodi della rete a spendere risorse per tale operazione nel sistema Bitcoin è stato pensato un premio per chi riesce a minare un blocco. Il valore di tale ricompensa, fissato a 50 BTC al momento della creazione del sistema, viene dimezzato ogni 210.000 blocchi (indicativamente ogni 4 anni). L'ultimo dimezzamento è avvenuto il 9 Luglio 2016 portando la ricompensa a 12.5 BTC. Attraverso questo processo viene coniata nuova moneta ed essendo l'unico modo per crearne di nuova è del tutto prevedibile la quantità di *coins* in circolazione.

### 2.2.1 Consenso decentralizzato

Come abbiamo già detto precedentemente, non vi è nessuna autorità garante del processo di validazione delle transazione e forse questa è la caratteristica più innovativa introdotta da Nakamoto quando presentò questo sistema alla comunità. Il consenso decentralizzato di Bitcoin avviene tramite quattro processi indipendenti:

- Ogni transazione è verificata indipendentemente dalle altre, da ogni nodo
- Ogni blocco è costruito in maniera indipendente dai *miners* insieme alla dimostrazione di avvenuta computazione attuata tramite il PoW (Capitolo 3)
- Ogni nodo verifica indipendentemente un blocco e lo inserisce nella catena

- Ogni nodo seleziona la catena più lunga (con più blocchi) in maniera indipendente. La catena più lunga ci garantisce difatti maggiore sicurezza da attacchi malevoli (Capitolo 3).

# Capitolo 3

# Proof-of-Work

La Proof-of-Work (in breve, PoW) è una tecnica che impiega risorse computazionali per certificare il lavoro compiuto da un utente. È stata introdotta da Adam Back [19] ed era stata inizialmente pensata per evitare attacchi di tipo Denial of Service (DoS).

Una Proof-Of-Work è un'algoritmo che impone di risolvere un problema computazionalmente complesso per dimostrare di aver impiegato delle risorse. Ogni Proof-Of-Work

- Deve essere asimmetrica, cioè deve essere computazionalmente difficile trovare la soluzione ma, fornita questa, computazionalmente facile verificarla.
- Deve permettere di poter aggiustare la difficoltà richiesta per la risoluzione del problema.
- Deve avere dei parametri che non permettono di avvantaggiarsi il lavoro anticipando la risoluzione del problema.

Anche se era stata inventata per evitare attacchi DoS, in Bitcoin viene impiegata per permettere di risolvere praticamente il problema del consenso distribuito.

## 3.1 Proof-of-Work in Bitcoin

Nei Bitcoin introdotti da Nakamoto [Macci, 2016] il metodo di Proof-of-Work utilizzato è basato sul calcolo della funzione hash crittografica *double-SHA256*. Dato un valore di difficoltà (sezione 2.1.2), occorre trovare in input un header del blocco per la funzione *double-SHA256* che restituisca un hash con $k$ zeri iniziali, dove $k$ è un intero in funzione della difficoltà. Chiaramente, essendo la funzione hash impiegata crittograficamente sicura, non è possibile prevedere l'output se non calcolandone il risultato e non è possibile prevedere il risultato con stime probabilistiche in quanto è distribuito

in maniera omogenea sul codominio. Per avere un numero di zeri iniziale pari al numero da noi cercato, dovremo quindi andare a calcolare l'hash modificando di volta in volta il valore dell'input. Tale modifica corrisponde alla modifica del *nonce* (sezione 2.1.2) che viene concatenato dell'header del blocco.

Supponiamo, per toccare con mano il problema, di voler creare un hash della stringa "Università di Pisa" che inizi con uno zero. Applichiamo la funzione SHA-256 trovando come valore hash

```
a4e3a0150b05841da09a76183fee0bca3d8f65580bc6818f05efb4696e7efb95
```

che come vediamo non rispetta la nostra condizione, cioè l'iniziare con uno zero.

Cerchiamo quindi, per tentativi, concatenando un intero $n$ che andremo ad incrementare fino al raggiungimento del nostro obiettivo, di trovare un input che soddisfi il nostro requisito.

| SHA256("Università di Pisa"+string(n)) | $n$ |
|---|---|
| fc2ad1c7f998b9fb82e71adf60e173550201aaaef1f4be6ec9607c02cd38e835 | 0 |
| a5d8e2f0c743d7166236e9115345cf39d6af8ff9927a364be7852d2bb750ce9d | 1 |
| c7fd7dd705a70fdd055ac9349fa0f36fa3840fee2fcd7e6086f832a454e7d0c9 | 2 |
| 893a8491ba02333c6aac8f20af10e56f64a4eeb4340383afcf82e88e9fd1609f | 3 |
| 2ecb5c464b50752210ba7ab21b3d796f58473a2b908dcff919319f78d36d99b0 | 4 |
| f3aa3755dcba9619d40d011bb24508125a4f40e95d51f06c7be51cc164afdaab | 5 |
| 0377e2c735b17670e99b6a470c1c85b2d0b9d5a714a98abb7cfe607537d67ebe | 6 |

Come possiamo vedere ci sono serviti 6 tentativi prima di trovare un risultato che fosse quello richiesto. In termini di probabilità, se l'output di una funzione hash è correttamente distribuito, ci aspettiamo di trovare un hash che inizia nuovamente con uno zero dopo 16 hash (essendo il nostro output in notazione esadecimale). Questa cosa, in termini numerici, vuol dire trovare un hash di valore più piccolo di

```
0x1000000000000000000000000000000000000000000000000000000000000000
```

che corrisponde in notazione decimale al valore $2^{256}$. Andando ad incrementare quindi il valore di zeri iniziali, andremo a richiedere di trovare hash sempre più piccoli numericamente e ciò sarà quindi più complesso. Generalizzando, dato il vincolo di $k$ zeri iniziali, occorreranno circa $16^k$ tentativi.

In Appendice è mostrato il codice di una possibile implementazione di tale algoritmo (sezione 5.4).

Chiaramente è semplice verificare che il *nonce* dichiarato sia quello che effettivamente permette di trovare il risultato ricercato in quanto l'operazione di verifica richiede tempo costante, basterà difatti prendere l'header del blocco, concatenare il *nonce* dichiarato e calcolarne l'hash.

## 3.2 Primecoin - Una Proof-of-Work "utile"

La Proof-of-Work basata sul calcolo di un hash non è la sola Proof-of-Work inventata. Sono stati proposti anche altri metodi che utilizzano sempre la complessità computazionale del calcolo di un risultato per certificare il lavoro svolto ma senza impiegare funzioni hash. Uno di questi altri metodi si basa sul calcolo dei numeri primi, ideato a Marzo 2013 da una o più persone che rispondono allo pseudonimo di Sunny King [5], viene attualmente usato nella criptomoneta **Primecoin**.

### 3.2.1 I numeri primi

I numeri primi sono sempre stata una costante della crittografia moderna. Il loro studio ha sempre affascinato l'uomo tant'è che sono state create delle categorie per evidenziare dei tipi particolari di primi che rispondono a determinate caratteristiche.

- Esistono i *primi gemelli*, dove $p$ e $p+2$ sono entrambi numeri primi.
- Esistono i *primi di Marsenne* che possono essere espressi nella forma $2^p - 1$.
- Ed esistono *primi di Sophie Germain* dove sia $p$ che $2p+1$ sono primi.

Estendendo il concetto dei primi di Sophie Gemain è stata definita una catena di coppie di primi vicini chiamata *catena di Allan Cunningham* (1842-1928) di cui ne esistono tre tipi:

#### Catena di Cunningham del primo tipo

Una catena di Cunningham del primo tipo di lunghezza $n$ è una sequenza di numeri primi $(p_1, \ldots, p_n)$ tale che per ogni $1 \leq i < n$, $p_{i+1} = 2p_i - 1$. La prima catena di questo tipo di lunghezza 5 che si trova è la seguente:

$$1531, 3061, 6121, 12241, 24481$$

#### Catena di Cunningham del secondo tipo

Una catena di Cunningham del secondo tipo di lunghezza $n$ è una sequenza di numeri primi $(p_1, \ldots, p_n)$ tale che per ogni $1 \leq i < n$, $p_{i+1} = 2p_i + 1$

#### Bi-twin chain

Una catena bi-twin di lunghezza $k+1$ (dove una coppia di primi compone un elemento) è una sequenza di numeri naturali

$$n-1, n+1, 2n-1, 2n+1, \ldots, 2^k n - 1, 2^k n + 1$$

Si noti che prendendo solo i numeri con la somma $(n+1,\ 2n+1,\ ...)$ si trova una Catena di Cunningham di secondo tipo, mentre prendendo solo quelli con la sottrazione si trova una catena di Cunningham di primo tipo.

Vediamo un esempio riportato direttamente da Sunny King [5] per capire meglio una catena bi-twin.

- Prendiamo la coppia di primi gemelli $(5, 7)$
- Al centro vi è 6. Raddoppiamo il 6, arrivando al 12
- Intorno a 12 c'è $(11, 13)$, che sono anch'essi primi gemelli

quindi la catena 5, 7, 11, 13 è una catena *bi-twin* di lunghezza 2, anche detta catena *bi-twin* con un solo collegamento.

### 3.2.2 Proof-of-Work

Come si è già detto, per essere efficace una tecnica di PoW deve fornire gli strumenti per far verificare facilmente il risultato da tutti i nodi della rete. Questo vuol dire che i numeri primi coinvolti non devono essere eccessivamente grandi, come i primi di Marsenne. Per questa ragione nel Proof-Of-Work di Primecoin sono accettati i tre tipi di catene di primi di cui si è parlato precedentemente (sottosezione 3.2.1).

Per verificare la correttezza dei risultati sottomessi viene usato il test di Fermat (sezione 5.2) insieme al test di Eulero-Lagrange-Lifchitz (sezione 5.3) per assicurarsi della primalità della catena [15].

#### Difficoltà

Altra caratteristica del PoW è la difficoltà calcolata dinamicamente nel tempo, permettendo alla moneta di tenere costante il tempo impiegato nel processo di *mining* indipendentemente dal sistema su cui viene eseguito. Inizialmente si potrebbe pensare di usare la lunghezza della catena come indicatore della difficoltà ma questo non è fattibile, difatti una catena lunga otto primi potrebbe essere centinaia di volte più difficile da trovare rispetto ad una di sette. Si potrebbe quindi pensare di usare come indicatore di difficoltà la grandezza dei numeri primi ma ciò complicherebbe la fase di verifica che invece vogliamo possa essere fatta velocemente. La soluzione per la verifica della primalità adottata è quella in cui si impiega il Test di Fermat in combinazione con il test di Eulero-Lagrange-Lifchitz . Il Test di Fermat (quelo di Eulero-Lagrange-Lifchitz lavora in maniera molto simile) è un modo veloce per dire se un numero è (con molta probabilità) primo. Volendo ad esempio dimostrare che $n = 17$ è primo, ci basterà eseguire il calcolo $2^n \mod n = 2$

e controllare che l'uguaglianza sia verificata. Se è corretta allora il numero $n$ è primo[1]. Difatti $2^{17} - 17 \cdot 7710 = 2$.

Abbiamo quindi ora un metodo per verificare la primalità dei numeri in una catena ma dobbiamo riuscire a modulare la difficoltà meglio di come abbiamo fatto. Per evitare di usare la lunghezza della catena, che come abbiamo detto ci discretizza la funzione difficoltà con degli intervalli troppo elevati, si è pensato di combinare questa con il resto del Test di Fermat. Sarà possibile quindi distinguere una catena lunga 7.2 da una lunga 7.5. Per fare ciò si cerca semplicemente il primo valore nella catena che non è primo. Più è piccolo, maggiore è la parte frazionaria. Per esempio, prendendo la catena

$$2, 5, 11, 23, 47$$

ha il valore successivo uguale a 95. Prendendo ora $2^{94} \mod 95 = 54$ potremo calcolare facilmente la lunghezza che risulterà

$$5 + \frac{(95 - 54)}{95} \cong 5.43$$

Introdotto quindi questo concetto, è stata definita in Primecoin la correttezza del PoW rispetto alle catene, queste infatti devono avere una lunghezza grande almeno quanto la difficoltà (a metà Luglio 2016, il valore richiesto è intorno ai 10.2).

#### Non riusabilità

Un altro fondamentale requisito di una PoW è la non riusabilità di risultati ottenuti in precedenza. Nei Primecoin tale condizione viene garantita imponendo che l'*origine*[2] della catena sia divisibile per l'hash del blocco che si vuole verificare. L'unico modo, quindi, di generare un blocco valido è di cercare una catena in riferimento ad un hash che si conosce e tale catena sarà utile solamente per la verifica di quel blocco.

## 3.3 Proof-of-Stake

La Proof-of-Stake è un metodo di verifica alternativo alla Proof-of-Work. Non viene difatti più usata l'idea di dover dimostrare di aver svolto dei calcoli fornendo un risultato computazionalmente complesso da calcolare bensì viene introdotto un nuovo concetto, la *coin age*, che serve per dimostrare di avere interesse nella gestione della moneta e acquisire quindi la fiducia della comunità per la fase di *mining*. La **coin age** è facile da calcolare: se

[1] In realtà il test di Fermat ci dice che un numero primo rispetta quella proprietà ma non ci dice nulla sui numero non primi. Esistono di fatti alcuni numeri, detti pseudorimi di Fermat, che soddisfano quella congruenza senza essere effettivamente primi

[2] L'origine è definita come la media dei primi due elementi.

Bob ha ricevuto 10 coin da Alice e li ha posseduti per 90 giorni, Bob ha 900 coin-days di *coin age*. Quando Bob spenderà quei 10 coin ricevuti da Alice, la *coin age* accumulata da Bob con quei coin sarà detta *consumata* (o *distrutta*) [4].

### 3.3.1 Motivazioni

L'idea di Nakamoto [Macci, 2016] per l'impiego del Proof-Of-Work ha permesso di gestire in maniera efficace l'intero sistema, tuttavia alla base di tutto vi è una dipendenza energetica che introduce un importante costo economico. Da questa osservazione nasce quindi l'idea di questa verifica alternativa, per dimostrare che una moneta p2p decentralizzata, come è quella di Bitcoin, per vivere non ha necessità di dipendere da un consumo energetico.

Nella Proof-of-Stake l'idea di base è quella di dimostrare di possedere la moneta (e di poterla quindi amministrare), da qui il nome, **Proof-of-Stake**, che è possibile tradurre in italiano come *prova che si ha una posta in gioco*.

### 3.3.2 Scelta del nodo successivo

Come nel Proof-of-Work, questo metodo di verifica è usato per decidere quale blocco successivo della Blockchain convalidare e inserire quindi in maniera definitiva nella catena. Nel PoW il criterio usato per concordare quale blocco, e quindi quale catena portare avanti, è quello per cui viene scelto il blocco con più lavoro computazionale. Nel PoS la scelta ricade su vari fattori:

- **Selezione Casuale**: Nei Blackcoin [13] viene utilizzata una funzione randomizzata per predire il generatore del blocco successivo. Tale valore è influenzato da una formula che cerca il valore hash più basso rapportato alla dimensione della somma in gioco. Essendo quest'ultimo un valore pubblico, è facile fare previsioni su quale account si aggiudicherà il diritto di minare il blocco successivo.

- **Selezione basata sulle Velocità**: Per evitare di far accumulare agli utenti grandi quantità di monete per aumentare il proprio patrimonio di *coin age*, alcune monete come i Reddcoin [14] scelgono il minatore successivo in base alla velocità di movimentazione, incoraggiando quindi allo scambio di moneta.

- **Selezione basata sul voto**: A differenza di altre monete che si basano su funzioni del tutto estranee ad interventi umani diretti, alcune monete come i BitShares hanno implementato un sistema che permette agli utenti di votare e scegliere chi far diventare il minatore successivo. [9]

- **Selezione basata sull'anzianità**: Nei PPCoin è stato introdotto, affiancato ad una selezione casuale, il concetto di anzianità. L'anzianità viene misurata in *coin age*, valore calcolato considerando solamente monete non spese per almeno 30 giorni e non più di 90, cosa che permette di non avere utenti con un anzianità tale da dominare sugli altri. Ad ogni *mining* il minatore consuma la quantità di *coin age*, cosa che garantisce che non sia sempre lo stesso a dominare la blockchain. [4]

### 3.3.3 Generazione dei blocchi nei PPCoin

Mostriamo ora un algoritmo di *mining* differente da quello impiego nei Bitcoin. Prendiamo quindi i **PPCoin** [4] per studiarne il processo di *mining* e vedere come viene usata la *coin age* per il calcolo del certificato.

- Per prima cosa occorre scegliere un output ricevuto tra gli ultimi 30 e 90 giorni.
- Occorre ora formare una struttura chiamata *kernel* che rappresenta una sorta di header del blocco. Conterrà l'output scelto, il tempo corrente e un valore chiamato *nStakeModifier* (un blocco di bit casuali periodicamente ricalcolato)
- Occorre ora calcolare l'hash del *kernel* in modo tale da verificare la condizione richiesta (similmente al PoW) dove la tale condizione è in funzione
  - della difficoltà derivante dal network (complessità maggiore ci impone un target più basso, sezione 2.1.2)
  - dell'età e della somma dell'output scelto, cioè quantità di valore in *age coin* che l'output ha generato (un output più "vecchio" ci permette di alzare il target, rendendo più semplice il calcolo dell'hash)[3].
- Se l'hash calcolato è più grande del target bisogna ritornare al primo punto
- Se invece l'output è inferiore al target, noi spendiamo questo output nella transazione *coinbase*, una transazione speciale che segna la *coin age* di quella transazione consumata, aggiungiamo la ricompensa e le commissioni sulle transazioni al blocco e firmiamo il tutto usando la chiave relativa all'output speso, dimostrando di essere quindi i proprietari di quella transazione

[3]Una semplice formula impiegata è spesso $proofhash < coinage \cdot target$, da cui è chiaro che alzando la coinage è possibile fornire un hash di valore maggiore e quindi più semplice da trovare.

- Il blocco è quindi stato creato e si può pensare al successivo

Ai lettori più attenti non sarà certo sfuggito che anche in questo algoritmo viene impiegato l'uso di hash per la ricerca di una soluzione prestabilita. Una sostanziale differenza dal PoW è che in questo caso le operazioni di hashing devono portare a trovare una soluzione molto meno restringente rispetto a quanto avviene nei Bitcoin dove la difficoltà calcolata dall'*hash rate* della rete ci impone vincoli molto più complessi da soddisfare (sezione 3.1).

Il target dell'hash è un valore variabile dipendente dalla *coin age* consumata negli output inseriti nel kernel, contrariamente a quanto avviane nel PoW dei Bitcoin dove il target è univoco e dipende dall'hash rate totale del network. Perciò per come viene calcolato, più *coin age* sono consumati nel kernel, più facile sarà calcolare un hash che soddisfi il target. [4]

### 3.3.4 Problematiche

Come è stato notato [6] questo metodo di verifica presenta alcune difficoltà. Il problema più grande è dovuto al fatto che è possibile, teoricamente, effettuare transazioni con la stessa moneta impiegando una quantità di risorse limitata. Nel caso di una ramificazione della Blockchain una persona può "votare" (cioè cercare di tenere in vita) per entrambe le varianti (avendo magari in entrambe delle transazioni che a lui risulterebbero vantaggiose). A differenza della PoW, questa volta però, continuare a lavorare su due rami della Blockchain non è così dispendioso poichè in caso di fallimento non vi è perdita (come nel caso di Bitcoin in cui è stata investita una grande quantità di energia). Si può tentare quindi di ingannare, senza dover investire troppe risorse, l'intero network e magari far risultare convalidate (per un periodo sufficiente di tempo) due transazioni che risulteranno, al ricongiungimento dei rami, duplicate. Per risolvere questo problema sono stati provati alcuni sistemi:

- I **Peercoin** utilizzano dei checkpoint trasmessi da un ente centrale, firmati con la chiave privata delle sviluppatore del sistema. Il problema di questa soluzione è che si è persa la completa decentralizzazione della blockchain. [4]

- È stato introdotto un protocollo sperimentale in una blockchain di nuova generazione chiamata **Nxt** in cui è possibile modificare solo gli ultimi 720 blocchi [16]. Tale valore è stato individuato poichè si è notato che era necessaria la modifica di almeno il doppio dei blocchi per riuscire a portare a termine l'attacco.

- Un altro protocollo, introdotto da **Ethereum**, con nome Slasher [17] prevede la possibilità di punire un utente che cerca di minare più blockchain contemporaneamente, andando quindi a limitare il potere

del singolo utente e quindi limitare la possibilità di attacchi sopra descritti

### 3.3.5 Tirando le somme

Chiaramente questo metodo è ancora in fase di studio e non è possibile ora dire se sostituirà o meno il Proof-of-Work. La prima moneta che ha usato la Proof-of-Stake come metodo di verifica principale è stata la PPCoin, sulla cui scia sono sorte molte altre monete che hanno saputo usare questo sistema combinandolo con il classico PoW anche nei modi più impensabili, come i BlackCoin che nel primo momento di vita hanno usato la PoW per poi passare esclusivamente alla PoS. Al giorno d'oggi sono poche le monete che hanno implementato la Proof-Of-Stake in maniera pulita, come ad esempio la criptomoneta Nxt che però ha numerosi problemi di sicurezza su cui sta ancora cercando una soluzione efficace.

## 3.4 Memory-Hard mining puzzle - Litecoin

Come si è già visto, il network dei Bitcoin sfrutta la funzione SHA256 per effettuare il Proof-of-Work. Il problema principale di tale tecnica è la possibilità di parallelizzare il processo in maniera incontrollata andando quindi a ridurre drasticamente il tempo impiegato per il processo con l'effetto di un aumento non indifferente della difficoltà (come si è già visto nella sezione 2.1.2) che comporta una centralizzazione del processo di *mining* divenuto troppo dispendioso per l'utente medio. Per ovviare a questo problema si è quindi pensato di cercare di costruire algoritmi di Proof-of-Work basati sulla velocità di accesso alla memoria invece che alla potenza di calcolo delle macchine. È nata così una moneta che ha cercato di implementare proprio quest'idea: i **Litecoin**. Questa Altcoin [4] utilizza un algoritmo chiamato *scrypt* che sostituisce la funzione SHA256 nel PoW. L'esecuzione di Scrypt, pur utilizzando all'interno la funzione SHA256, dipende soprattutto dall'accesso ad una grande quantità di memoria in cui vengono salvati risultati temporanei (e non dalla velocità computazionale) cosa che rende complicato riuscire a far eseguire molte istanze in parallelo. Per la sua struttura risulta anche più difficoltoso costruire hardware specializzato. Queste caratteristiche si stanno rivelando vantaggiose in quanto non sembra che la procedura di *mining* si stia centralizzando come invece è accaduto con i Bitcoin.

### 3.4.1 Algoritmo di hash dei blocchi

Come nel PoW dei Bitcoin, quando si effettua il *mining* dei Litecoin, viene calcolato moltissime volte l'hash dell'header del blocco. Tale header presenta la stessa struttura di Bitcoin. Il corpo del blocco contiene tutte le

[4] Termine con cui si indicano tutte le monete alternative ai Bitcoin

transazioni, salvate tramite l'albero di Merkle indirettamente nell'header. La struttura generale è identica a quella di Bitcoin se non per l'utilizzo della funzione hash **Scrypt**.

### 3.4.2 Scrypt

Scrypt è una funzione hash creata in originale per il servizio di backup *Tarsnap online backup service* [5]. Ha i seguenti parametri:

- `d` - Il dato da derivare
- `salt` - Un valore da concatenare al dato
- `N` - Il carico computazionale della CPU
- `r` - Il carico della Memoria
- `p` - Il parametro di parallelizzazione
- `dkLen` - La lunghezza dell'output

Tali parametri possono essere decisi dall'utente in base alle risorse disponibili sulla macchina su cui si esegue la funzione. I valori usati nei Litecoin sono:

- salt = gli 80 bytes dell'header del blocco
- N = 1024
- r = 1
- p = 1
- dkLen = 256 bits

Per approfondire l'implementazione si faccia riferimento al documento preparatorio per un RFC mai divenuto tale [12].

### 3.4.3 Tirando le somme

Ad oggi qualsiasi computer può effettuare il *mining* dei Litecoin, cosa che non è più sostanzialmente possibile con i Bitcoin data la potenza di calcolo di pochi nodi che dominano il processo di *mining*. Inoltre ogni processore può minare Litecoin, cosa che è non sostanzialmente possibile con i Bitcoin data la potenza di calcolo dei computer con cui si compete. Come nei Bitcoin l'aggiornamento della difficoltà avviene ogni 2016 blocchi tuttavia i Litecoin generando un blocco ogni 2.5 minuti ricalcolano la difficoltà ogni 3.5 giorni. Questa maggiore frequenza rende l'intero sistema più decentralizzato e veloce. Un commerciante può inoltre verificare una transazione in un quarto

[5] `http://www.tarsnap.com`

del tempo che occorrerebbe per verificare una transazione in Bitcoin, cosa che riduce il rischio di avere un attacco di tipo double-spending. Tuttavia i Litecoin non sono riusciti a risolvere il problema del Time warp Attack (cioè cercare di anticipare il *mining* di blocchi successivi inserendo transazioni malevole e impedendo agli altri di accorgersi del problema in tempi utili) che rimane comunque praticabile con sufficienti risorse [6].

[6] `https://litecoin.info/Time_warp_attack`

# Capitolo 4

# Usi alternativi della Blockchain

Dopo aver visto nel precedente capitolo come poter effettuare il *mining* diversamente dallo stile classico introdotto in Bitcoin, andremo in questo capitolo a cercare soluzioni e utilizzi alternativi alla Blockchain, una tecnologia che secondo molti sopravviverà a Bitcoin che invece sono destinati a perdere di interesse.

## 4.1 Metodi per il timestamping

Il primo utilizzo alternativo che andiamo ad analizzare per la Blockchain è il timestamping. In fin dei conti la Blockchain può essere vista come un database dove poter inserire dei record che, una volta scritti, non è possibile modificare. Pensiamo ad un caso pratico: voglio dimostrare di conoscere un determinato dato prima di una data. In passato avrei avuto diversi modi per certificare questa cosa:

- avrei potuto spedire a me stesso una cartolina, usando poi la data del timbro postale come prova del fatto che quel dato esisteva già in un determinato giorno
- avrei potuto acquistare uno spazio pubblicitario su un giornale andando quindi ad associare la data di creazione del dato con la data di pubblicazione del quotidiano
- avrei potuto scrivere il dato su un foglio e farlo firmare da una persona certificata (come un notaio) ed usare quindi il foglio firmato come prova di esistenza del dato

Ad oggi, invece, con la Blockchain sono in grado di dimostrare che possiedo un determinato dato senza dover ricorrere a trucchi particolari. Mi basterà inserire nella Blockchain il dato e questo sarà salvato per sempre, replicato

su migliaia di macchine in giro per il mondo. Non resta quindi che capire come poter salvare questo dato nella Blockchain, andando di fatto a piegare lo strumento per uno scopo diverso da quello a cui si era pensato. A tal proposito vi è difatti un grande dibattito tra gli utenti della rete di bitcoin che sono divisi in due scuole di pensiero, una che promuove l'utilizzo della Blockchain per scopi laterali come questo appena descritto, l'altra che invece scoraggia scopi diversi dalla registrazione di transazioni accusando di far aumentare senza reale necessità la dimensione della Blockchain che già a Maggio 2016 era di circa 63 GB.

Prima di analizzare i metodi di timestamping ho intenzione di riportare una curiosità sul perchè serve uno strumento informatico tanto complesso come la Blockchain quando esistono altri metodi in informatica che ci mostrano la data di creazione del file. Non mi soffermerò neanche un minuto a discutere sull'inaffidabilità della data di creazione del file mostrata dal sistema, facilmente modificabile anche da utenti non esperti. È già più interessante invece pensare a metodi come quello descritto prima in relazione al passato: una terza persona che ti firma il messaggio certificandone l'ora di produzione, come potrebbe essere la data di pubblicazione di un post su un social network. Ed eccolo qui l'aneddoto! Qualche tempo fa un simpatico signore ha pensato di voler dimostrare che le partite della Fifa fossero truccate e per farlo ha ben pensato di creare un account Twitter chiamato Fifa Corruption. L'idea geniale qual è stata? Ha scritto un tweet per ogni possibile esito di ogni partita della giornata e, terminati i match, ha rimosso tutti quei tweet che non si erano avverati volendo di fatto dimostrare, tramite la data di pubblicazione dei tweet, di conoscere in anticipo il risultato della partite. Un piano geniale rovinato da un semplice screenshot (Figura 4.1) che ha evidenziato l'inganno.

Chiaramente la Blockchain non ci aiuterebbe a risolvere questo problema a meno di non pubblicare in chiaro i dati, cosa che non è sempre possibile in quanto la quantità di dati scrivibili in una transazioni è di pochi bytes. Escluso quindi l'utilizzo per dimostrare di prevedere il futuro, con la Blockchain potrei dimostrare di conoscere un messaggio in una certa data, senza mostrare direttamente quel messaggio fino a che non occorra esibire la prova di conoscenza, semplicemente salvando l'hash del messaggio in una transazione.

Per non scontentare eccessivamente gli utenti che ne scoraggiano l'uso diverso dal salvataggio di transazioni [18], andremo a salvare solamente un hash del dato finale (che normalmente è più piccolo del dato stesso).

### 4.1.1 Metodo 1 - Pagamento all'*hash*

Il primo metodo, quello più semplice che ci può venire in mente è il seguente. Preso l'hash effettuo un pagamento simbolico ad esso come se fosse una chiave pubblica di un qualche utente. Chiaramente tale procedura farà per-


Figura 4.1: L'account FIFA Corruption prima della cancellazione dei tweet

dere immediatamente il possesso dei coins inviati che vengono detti *bruciati* in quanto nessuno potrà più spenderli (nessuno difatti conosce la "chiave privata" associata all'hash in questione). L'idea di bruciare soldi non è proprio delle migliori e inoltre i *miners* non hanno modo di sapere che quella transazione non sarà mai spesa e così sarà tenuta in eterno nell'UXTO (sottosezione 1.1.1).

### 4.1.2 Metodo 2 - Pagamento all'*hash* 2.0

Questo secondo metodo, chiamato *CommitCoin*, è un po' più elaborato del primo ma ci permette di non bruciare *coins*. Prendiamo il nostro hash e consideriamolo come chiave privata. Calcoliamo quindi la chiave pubblica associata. Inviamo ora una piccola transazione (esempio 20000 satoshi) all'indirizzo pubblico trovato e rimandiamo indietro questi soldi in due transazioni distinte (esempio 10000 satoshi l'una). Ora arriva la parte cruciale: scegliamo lo stesso numero random per effettuare l'ECDSA (Elliptic Curve

Digital Signature Algorithm). Questa nostra "mancanza", l'aver scelto cioè due numeri random identici [1], permetterà ad un utente di calcolare facilmente la chiave privata che altro non sarà che l'hash del dato che volevamo mostrare.

Riportiamo ora il metodo completo per capire come è stato effettuato l'attacco. Per prima cosa occorre ricordarsi le formule necessarie per l'esecuzione dell'algoritmo ECDSA:

$$R = k * G$$

dove R rappresenta un punto della curva e G il punto base della curva, un generatore della curva con un ordine elevato

$$S = k^{-1}(z + rd_A) \mod n$$

Dove

- $z$ sono i bits sono i primi $L_n$ bits dell'hash del dato da firmare, con $L_n$ la lunghezza in bit dell'ordine della curva
- $r = x_1 \mod n$
- $d_A$ rappresenta la chiave privata

Analizziamo ora la seconda equazione. Questa ha due parametri sconosciuti, $k$ e $d_A$. Appare chiaro che la sicurezza si basa quindi sul tenere $d_A$ privata e calcolare $k$ con una funzione casuale crittograficamente sicura. Se infatti si conosce anche solo uno di questi due valori, ci ritroviamo una semplice equazione di primo grado da risolvere.

Vediamo ora che succede andando a firmare due hash con lo stesso $k$. Avremo quindi ora

$$S = k^{-1}(z + rd_A)$$
$$S' = k^{-1}(z' + rd_A)$$

Andiamo a sottrarre le due firme

$$S - S' = k^{-1}(z + rd_A) - k^{-1}(z' + rd_A) = k^{-1}(z + rd_A - z' + rd_A) = k^{-1}(z - z')$$

Cioè

$$k = \frac{z - z'}{S - S'}$$

Abbiamo così trovato uno dei due parametri che dovevamo tenere nascosti per evitare di dare all'attaccante una semplice equazione di primo grado. Il calcolo di $d_A$ risulta quindi banale.

[1] Questa mancanza è stata proprio la causa che ha permesso di portare a termine l'attacco alla Sony nel 2010, permettendo ad un gruppo di persone esterne all'azienda di firmare a nome della Sony alcuni software per PlayStation 3. Per approfondire la necessità della scelta casuale del valore in questione si lascia il riferimento all'RFC 6979 `https://tools.ietf.org/html/rfc6979`

### 4.1.3 Metodo 3 - L'OP_RETURN

Questo è il metodo più complesso. Sfrutta l'OP_RETURN, una particolare istruzione che permette agli script (istruzioni che si possono concatenare alle transazioni e vengono eseguite per la validazione della transazione) di impostare un valore da restituire al momento dalla verifica della transazione. Tale transazione verrà quindi impressa nella Blockchain durante il *mining* e permetterà di verificare che quel dato esisteva in quel momento. Effettuare questa operazione non è banale e quindi sono nati diversi servizi online che assistono nell'operazione. Il sito `http://coinsecrets.org` raccoglie tutti gli OP_RETURN inseriti da questi servizi. Uno di questi servizi, tanto per citarne uno, è `https://proofofexistence.com` che permette di salvare l'hash di un file con questo metodo dietro pagamento di 5 mBTC (con il cambio attuale circa 2 EUR).

## 4.2 DNS Distribuito

Un altro utilizzo interessante della Blockchain è l'uso come database di certificati di proprietà di beni durevoli nel tempo. Immaginando sempre la Blockchain come un database non modificabile, possiamo trovare metodi per salvare il certificato di proprietà di beni a favore di persone fisiche. Facendo così non ci sarebbero più problemi quali attacchi o perdite di dati come può accadere nel caso di database centralizzati e inoltre, per massima trasparenze, tutti i dati sarebbero fruibili liberamente da ogni utente. Anche se l'idea può essere applicata a beni fisici come auto e immobili, per ora concentriamoci su beni virtuali come può essere un dominio internet.

Un utente può liberamente acquistare un dominio internet recandosi dal registro di quel dominio e chiedendone l'assegnazione alla sua persona, acquisendo di fatto la possibilità di modificare il dominio *puntantolo* all'indirizzo IP scelto. Tutto il sistema ad oggi viene gestito tramite una gerarchia di server andando però a centralizzare l'intero database per ogni singola estensione [2]. Cercando quindi di realizzare un DNS libero e distribuito è nato Namecoin.

### 4.2.1 Namecoin

**Namecoin** (NMC) nascono come fork di Bitcoin. Non è interessante analizzarne quindi il funzionamento in quanto utilizzano gli stessi strumenti dei Bitcoin, dal PoW al sistema di ricompensa che ne limita il numero massimo prodotto a 21 milioni. La principale differenza con i Bitcoin è la possibilità di salvare dati nella Blockchain. Inizialmente si era pensato di fornire

[2]Sono state cercate soluzioni per migliorare l'efficienza ed evitare di centralizzare l'intero sistema, utilizzando di fatto proxy e DNS cache, ma sono implementazioni che trascuriamo

la possibilità di salvare dati in maniera diretta tuttavia nell'ultima implementazione, viste le difficoltà nel rendere scalabile il sistema, si è scelto di permettere di inserire contenuti nella blockchain solamente legandoli ad una transazione.

Il dominio di primo livello che è stato creato è il `.bit`, che ha funzioni similari a quelle del `.com` ma non dipende dall'ICANN[3].

Ogni record nei Namecoin consiste in una coppia chiave-valore. Ad esempio la chiave `d/example` rappresenta un record DNS nel *namespace* `d` e con nome `example`, corrispondente all'indirizzo `example.bit`. Il *namespace* `d` rappresenta che il valore seguente è un dominio. Il protocollo dei Namecoin permette difatti di inserire qualsiasi tipo di dato definendo a priori un *namespeace*.

Per inserire un dato occorre effettuare una transazione del valore minimo di 0.01 NMC. Tale dato sarà considerato valido per i successivi 36000 blocchi, andando ad evitare che i domini siano registrati e dimenticati e se ne perda per sempre la possibilità di recupero in caso il proprietario lo faccia cadere in disuso.

[3]Internet Corporation for Assigned Names and Numbers: `https://en.wikipedia.org/wiki/ICANN`

# Capitolo 5

# Appendice

Per approfondire le funzioni hash si rimanda al testo [2] e al testo [3].
Per la crittografia a chiave pubblica e la crittografia su curve ellittiche si rimanda al testo [3].

## 5.1 Albero di Merkle

L'albero di Merkle è un albero in cui ogni nodo non foglia è etichettato usando un hash generato dalla concatenazione delle etichette dei figli. Questa struttura dati fu brevettata da Ralph Merkle nel 1979 [7] [8]. La sua caratteristica principale è quella di riuscire a riassumere molti dati in un'unica struttura sulla quale è possibile fare delle operazioni molto efficienti tra le quali verificare che un determinato dato sia presente nell'albero e di verificarne la sua integrità.

L'albero di Merkle è un albero binario completamente bilanciato costruito ricorsivamente dalle foglie concatenando di volta in volta gli hash dei due nodi figli.

### 5.1.1 Prova di inclusione

Una volta che $n$ foglie sono state inserite nell'albero tramite il loro hash, si può facilmente verificare l'inclusione con $O(\log_2 n)$ passi, cosa che rende questa struttura molto efficiente. Per dimostrare che un elemento $T$ è inclusa nell'albero, un utente deve fornire un *authentication path* Figura 5.2. Ogni altro utente potrà quindi verificare tale percorso calcolando un numero limitato di hash.

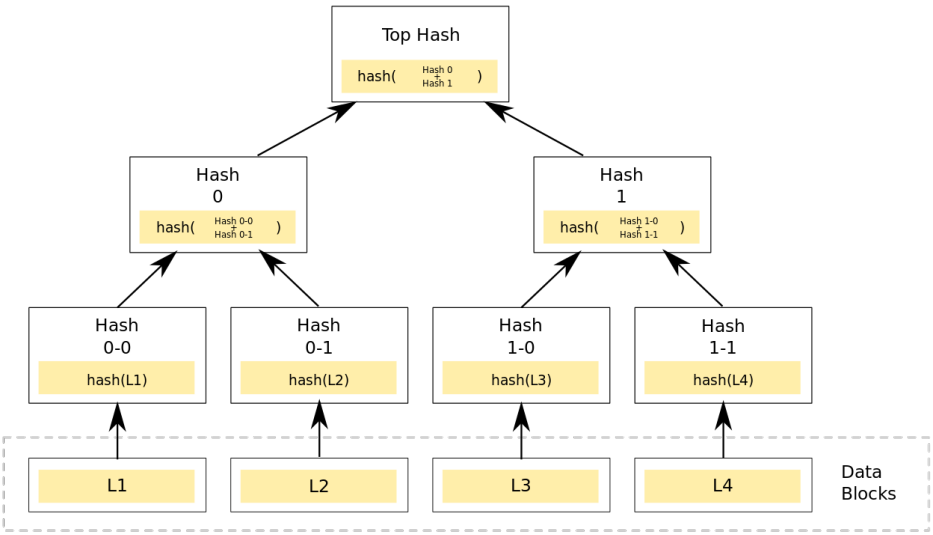



Figura 5.1: Esempio di un Albero di Merkle

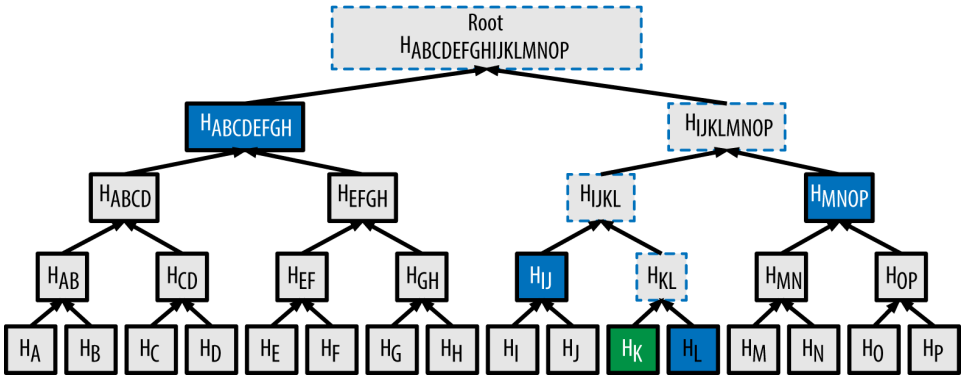



Figura 5.2: Authentication Path, raffigurato in blu, rispetto alla nodo $H_K$

## 5.2 Test di Fermat

Il test di Fermat è un test di primalità che utlizza il piccolo teorema di Fermat. Tramite questo test siamo in grado di dire con certezza che un numero $n$ non è primo. Dal piccolo teorema difatti sappiamo che se $p$ è un numero primo e $a$ è un numero intero che non è multiplo di $p$, allora vale

$$a^{p-1} \equiv 1(p)$$

Per un ulteriore approfondimento si rimanda a [10].

## 5.3 Test di Eulero-Lagrange-Lifchitz

Se $p \geq 5$ è primo, $q = 2p + 1$ è primo $\Leftrightarrow$ $q$ divide $3^p - 1$.
Si noti che $q$ è un numero primo di Sophie Germain.

Per un ulteriore approfondimento e per la dimostrazione si rimanda a [11].

## 5.4 Algoritmo semplificato del Proof-of-Work in Bitcoin

Questo algoritmo, riportato in [1] ci permette di capire facilmente come viene effettuato il Proof-Of-Work nei Bitcoin.

```
#!/usr/bin/env python
# example of proof-of-work algorithm

import hashlib
import time

max_nonce = 2 ** 32 # 4 billion

def proof_of_work(header, difficulty_bits):

    # calculate the difficulty target
    target = 2 ** (256-difficulty_bits)

    for nonce in xrange(max_nonce):
        hash_result = hashlib.sha256(str(header)
                                     +str(nonce)).hexdigest()

        # check if this is a valid result, below the target
        if long(hash_result, 16) < target:
            print "Success with nonce %d" % nonce
            print "Hash is %s" % hash_result
            return (hash_result,nonce)

    print "Failed after %d (max_nonce) tries" % nonce
    return nonce


if __name__ == '__main__':

    nonce = 0
    hash_result = ''

    # difficulty from 0 to 31 bits
    for difficulty_bits in xrange(32):

        difficulty = 2 ** difficulty_bits
        print "Difficulty: %ld (%d bits)"
                              % (difficulty, difficulty_bits)

        print "Starting search..."

        # checkpoint the current time
        start_time = time.time()
```

```
# make a new block which
#includes the hash from the previous block
# we fake a block of transactions - just a string
new_block = 'test block with transactions' + hash_result

# find a valid nonce for the new block
(hash_result, nonce) =
                proof_of_work(new_block, difficulty_bits)

# checkpoint how long it took to find a result
end_time = time.time()

elapsed_time = end_time - start_time
print "Elapsed Time: %.4f seconds" % elapsed_time

if elapsed_time > 0:

    # estimate the hashes per second
    hash_power = float(long(nonce)/elapsed_time)
    print "Hashing Power: %ld hashes per second" % hash_power
```

# Bibliografia


[Macci, 2016] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008, https://bitcoin.org/bitcoin.pdf

[1] Andreas M. Antonopoulos, Mastering Bitcoin, O'Reilly, 2014

[2] Cormen, Leiserson, Rivest, Stein, Introdution to Alghoritms, Third Edition, The MIT Press, 2009

[3] Anna Bernasconi, Paolo Ferragina, Fabrizio Luccio, Elementi di Crittografia, Pisa University Press, 2015

[4] Sunny King, Scott Nadal, PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, 19 Agosto 2012

[5] Sunny King, Primecoin: Cryptocurrency with Prime Number Proof-of-Work, 7 Luglio 2013

[6] Andrew Poelstra, Distributed Consensus from Proof of Stake is Impossible, 2015, https://download.wpsoftware.net/bitcoin/pos.pdf

[7] Ralf Merkle, "Method of providing digital signatures", published Jan 5, 1982, assigned to The Board Of Trustees Of The Leland Stanford Junior University http://bit.ly/245k8V1

[8] R. Merkle, A Digital Signature Based on a Conventional Encryption Function, Advances in Cryptology — CRYPTO '87. Lecture Notes in Computer Science 293. p. 369, http://bit.ly/1rkd19K

[9] Delegated Proof of Stake, BitShares, http://docs.bitshares.eu/bitshares/dpos.html

[10] Caldwell C., Finding primes & proving primality, 2002 http://primes.utm.edu/prove/

[11] Lifchitz H., Generalization of Euler-Lagrange theorem, 1998 http://www.primenumbers.net/Henri/us/NouvTh1us.htm

[12] The scrypt Password-Based Key Derivation Function, https://tools.ietf.org/html/draft-josefsson-scrypt-kdf-04

[13] Pavel Vasin, BlackCoin's Proof-of-Stake Protocol v2, http://bravenewcoin.com/assets/Whitepapers/blackcoin-pos-protocol-v2-whitepaper.pdf

[14] Larry Red, Proof of Stake Velocity: Building the Social Currency of the Digital Age, https://www.reddcoin.com/papers/PoSV.pdf

[15] Primecoin: the Cryptocurrency Whose Mining is Actually Useful, Vatalik Buterin, Bitcoin Magazine

[16] NXT: Whitepaper, http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt

[17] Slasher: A Punitive Proof-of-Stake Algorithm, Vitalik Buterin, https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/

[18] Developers Battle Over Bitcoin Block Chain, Danny Bradbury, http://www.coindesk.com/developers-battle-bitcoin-block-chain/

[19] Adam Back, Hashcash: Popular proof-of-work system, 1997